View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Electronic archive USPU

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Уральский государственный педагогический университет»
Институт математики, информатики и информационных технологий
Кафедра информационных технологий и методики обучения информатике

РАЗРАБОТКА ИГРОВОГО КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ

Выпускная квалификационная работа бакалавра
по направлению 02.03.02 - Фундаментальная информатика
и информационные технологии

Исполнитель: студент группы Б-41
института математики,
информатики и ИТ
Рямов И.С.

Руководитель: ассистент кафедры ИИТиМОИ
Алексеевский П.И.

Работа допущена к защите
«____»_____ 2017 г.
Зав. кафедрой _____

Екатеринбург – 2017

Реферат

Рямов И.С. РАЗРАБОТКА САЙТА ПОРТФОЛИО СТУДЕНТОВ, выпускная квалификационная работа: 35 стр., рис. 21, табл. 0, библ. 22 назв., приложений 6.

Ключевые слова: ИГРОВОЕ ПРИЛОЖЕНИЕ, СЕТЕВАЯ ИГРА, КЛИЕНТ-СЕРВЕРНОЕ ПРИЛОЖЕНИЕ.

Объект разработки – кроссплатформенная, сетевая, компьютерная игра.

Цель работы – разработать сетевую компьютерную игру, используя кроссплатформенные средства обеспечения клиент-серверного взаимодействия.

В работе описаны результаты проектирования и программной реализации сетевой компьютерной игры. Основное преимущество игр в том, что они являются многофункциональным инструментом в развитии логического мышления. Игровое приложение позволяет пользователю проанализировать, обобщить, систематизировать и активно развивать знания, умения, личностные качества.

Игровое приложение реализовано на сетевом уровне и выполнено с использованием языков JavaScript и Python, а также языка разметки документов HTML и каскадной таблицей стилей CSS.

Приложение прошло апробацию в компании «Тензор».

Оглавление

Введение

В настоящее время наблюдается активное развитие облачных технологий. Все больше программного обеспечения становится доступным посредством глобальных сетевых платформ и сервисов. Следовательно, нет необходимости в установке продуктов на компьютер и различные портативные устройства, т.к. для пользования услугами необходим лишь доступ в Интернет.

Компьютерные игры за время своего существования претерпели множество качественных изменений в части выразительности и технологичности. Наблюдается проникновение компьютерных игр в различные сферы деятельности – образование, искусство, спорт. В образовательной сфере, например, они могут служить инструментом для проведения лекций, экзаменов, профессиональной практики. Также игры могут помочь в таких вещах, как развитие пространственного мышления, улучшение зрительной реакции, снятие стресса, тренировка организаторских навыков [7]. Являясь одним из наиболее распространенных на рынке классов ПО, дальнейшее развитие компьютерных игр с использованием облачных технологий представляется закономерным.

Одно из направлений развития компьютерных игр – реализация его в форме веб-приложения.

Основное преимущество игрового взаимодействия заключается в том, что оно является многофункциональным инструментом в развитии логического мышления, внимания, памяти, реакции и других навыков. Игровое приложение позволяет пользователю проанализировать, обобщить, систематизировать и активно развивать знания, умения, личностные качества [7].

Объектом исследования в данной работе является разработка клиент-серверных приложений. **Предметом** исследования является разработка кроссплатформенного клиент-серверного приложения. **Цель** работы –

разработать сетевую компьютерную игру, используя кроссплатформенные средства обеспечения клиент-серверного взаимодействия.

Для достижения поставленной цели требуется решить следующие **задачи**:

1. Анализ подходов к разработке клиент-серверных приложений.
2. Выявление критериев отбора инструментов разработки.
3. Сравнительный анализ инструментов разработки на основании выделенных критериев.
4. Разработка и реализация кроссплатформенной сетевой компьютерной игры.

Глава 1. ТЕОРЕТИКО-АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1. Подходы к разработке клиент-серверного приложения

Разработка – это детальное проектирование с исследованием и реализацией: цель, задачи, концепция, коммуникации, архитектура, оценка ресурсов [4].

Выделяют следующие подходы [2, 4]:

1. **Нативный** (платформенно-зависимый) подход. Разработка приложения на «родном» языке для каждой платформы. Например, так разработаны те приложения, которые идут «вшитыми» в устроство – галерея, камера, заметки, сообщения.
2. **Кроссплатформенный** подход. Приложение разрабатывается всего один раз, предполагается развертывание на различных платформах.
3. **Гибридный** подход. Представляет собой объединение особенностей нативной и кроссплатформенной разработки. Когда речь идет о решении определенных задач, эффективным решением будет комбинирование двух предыдущих подходов.

Исходя из вышесказанного, может сложиться мнение о том, что кроссплатформенное приложение станет успешным на всех платформах, включая и самые непопулярные. Однако для достижения требуемых целей потребуется дополнительный код поддержки различных платформ. Платформенно-зависимый подход широко распространен, но сопряжен с рядом трудностей при возможном переносе программного продукта на альтернативные платформы. При этом существуют стандарты и рекомендации, правила и инструкции по использованию фирменного стиля, помогающие придать элементам интерфейса «привычные» расположение и внешний вид [5]. Языковая среда, в которой разрабатывается платформенно-зависимое приложение, как правило, имеет необходимые инструменты для реализации привычного интерфейса. Напротив, процесс создания кроссплатформенного приложения с использованием веб-технологий может требовать значительных затрат ресурсов. Решить данную проблему помогают

т.н. фреймворки – наборы библиотек и готовых программных модулей, позволяющих быстрее и проще создать структуру приложения [6]. Различные кроссплатформенные фреймворки (Kendo UI, Framework 7, Ionic и другие) позволяют реализовать интерфейс целевой платформы, однако производительность при использовании такого подхода будет ниже, чем при непосредственном использовании платформенно-зависимых механизмов.

Приложение, разработанное под конкретную программно-аппаратную платформу, обычно имеет прямой доступ к основным возможностям целевого устройства и его программным сервисам. При проектировании кроссплатформенного приложения потенциал проекта ограничивается возможностями используемого фреймворка. Зачастую кроссплатформенный фреймворк не имеет возможности использовать весь функционал каждой из поддерживаемых платформ, предоставляя программные интерфейсы для наиболее общих функций. Такой набор возможностей является достаточным для большинства проектов.

К достоинствам платформенно-зависимого подхода можно отнести:

1. Высокая производительность приложения ввиду отсутствия промежуточных кроссплатформенных интерфейсов.
2. Обычно более широкий спектр доступных возможностей устройства.
3. Тесная интеграция с пользовательским интерфейсом целевой платформы.

Основным недостатком платформенно-зависимого подхода является сравнительно высокая трудоемкость переноса ПО на альтернативные платформы.

Кроссплатформенный подход имеет следующие достоинства:

1. Низкая трудоемкость переноса ПО на новые и перспективные платформы.
2. Простота сопровождения программного продукта ввиду использования единых подходов применительно к различным платформам.

Недостатки кроссплатформенного подхода включают:

1. Сравнительно высокую трудоемкость разработки в связи с необходимостью обеспечения работоспособности на каждой из поддерживаемых платформ.
2. Снижение производительности конечного продукта из-за использования промежуточных программных интерфейсов платформенной абстракции.

В результате анализа различных подходов к разработке для реализации клиент-серверного игрового приложения был выбран кроссплатформенный подход.

1.2. Этапы разработки клиент-серверного приложения

Проектирование пользовательского интерфейса – важная составляющая всего проекта. На этапе проектирования легче сформулировать требования и ограничения, предъявляемые к конечному программному продукту. Выявление ошибок на этапе проектирования позволит снизить затраты на тестирование и отладку приложения. На этапе проектирования мы сможем узнать потребности целевой аудитории, что наиболее важно, а что нет. Мы сможем проанализировать и спроектировать поведение пользователей на сайте. Благодаря проектированию можно с наилучшим эффектом разделить работу над Интернет-ресурсом.

Процесс работы над проектом состоит из следующих этапов [3]:

1. Определение целей и задач проекта.
2. Разработка дизайна.
3. Верстка.
4. Программирование.
5. Тестирование.
6. Запуск и сопровождение.

Слово «игра» в Русском толковом словаре под ред. С.И.Ожегова и Н.Ю.Шведовой в одном из значений определено как занятие, служащее для развлечения, отдыха, спортивного соревнования.

Игровое приложение реализует способ взаимодействия пользователей на игровом уровне, демонстрирующий уровень их логического мышления.

По результатам анализа подходов к разработке процесс реализации данного проекта был разделен на следующие этапы [1, стр.151]:

1. Начальная фаза.
   1.1. Определение целей приложения.
   1.2. Определение целевой аудитории.
2. Уточнение.
   2.1. Формулирование требований к функционалу программного продукта.
   2.2. Выбор языка программирования и средств для разработки сервера.
3. Построение.
   3.1. Система контроля версий.
   3.2. Разработка дизайна клиентских компонентов и проектирование интерфейсов клиент-серверного взаимодействия.
   3.3. Реализация серверных компонентов и интерфейсов клиент-серверного взаимодействия. Сторонние библиотеки.
4. Внедрение.
   4.1. Тестирование.
   4.2. Апробация методом экспертных оценок.

Цель приложения. На этом этапе необходимо максимально точно определить цели создаваемого проекта.

Целевая аудитория. На данном этапе определяется целевая аудитория создаваемого приложения. От этого напрямую будут зависеть необходимый функционал и дизайн проекта.

Функционал проекта. Определяются возможности создаваемого проекта, основные функции.

Выбор языка программирования для веб-сервера. Выбор языка напрямую зависит от поставленных задач.

PHP – язык программирования, исполняемый на стороне веб-сервера. Достаточно гибкий и мощный, что объясняет большую популярность и использование в проектах любого масштаба [21].

Преимущества:

1. Свободное ПО (PHP license).
2. Простое освоение на всех этапах.
3. Поддержка со стороны большого сообщества пользователей и разработчиков.
4. Развитая поддержка баз данных.
5. Наличие огромного количества библиотек и расширений языка.
6. Возможность использования в изолированной среде.
7. Развертка почти на любом сервере.
8. Портирован во множество аппаратных платформ и ОС.

Недостатки:

1. Не подходит для создания десктопных приложений или компонентов системы.
2. Наличие слабых средств для работы с исключениями.
3. Глобальные параметры конфигурации влияют на базовый синтаксис языка, затрудняя тем самым настройку сервера и разворачивание приложений.
4. Проблемы с безопасностью у веб-приложений, написанных на PHP.

Python. Применяется как интерпретируемый язык для скриптов различного рода назначений. Синтаксис языка является дисциплинирующим и эффективным, что существенно облегчает программистам совместную работу над кодом. Python – мультипарадигмальный язык программирования: позволяет совместить процедурный подход к написанию кода с объектно-ориентированным и функциональным [8, 9, 11, 16].

Преимущества:

1. Прост в изучении, особенно – на начальных этапах.
2. Открытая разработка.
3. Синтаксис с его особенностями стимулируют писать хорошо читаемый код.
4. Предоставление средства динамической семантики и быстрого прототипирования.
5. Наличие большого сообщества, которое хорошо настроено по отношению к новичкам.
6. Огромное количество полезных библиотек и расширений языка с легким использованием в своих проектах – предельно унифицированный механизм импорта.
7. Абсолютно все является объектами в смысле ООП, при этом программисту не навязывается объектный подход.

Недостатки:

1. Не очень удобная поддержка многопоточности.
2. Не такое большое количество качественных программных проектов по сравнению с другими универсальными языками программирования, например, с Java.
3. Изначальная ограниченность средств по работе с базами данных.

Java является строго типизированным объектно-ориентированным языком программирования [13].

Преимущества:

1. Транслирование программы на Java в байт-код Java, который выполняется виртуальной машиной Java – полная независимость байт-кода от операционной системы и оборудования, что позволяет запускать Java-приложение любом устройстве, для которого есть виртуальная машина.
2. Гибкая система безопасности – исполнение программы полностью контролируется виртуальной машиной.

Недостатки:

1. Сравнительно большой объем потребляемой памяти.
2. Невысокая производительность в некоторых случаях.

Система контроля версий. Необходимость программного обеспечения для облегчения работы с изменяющейся информацией очень высокая. Достаточно часта ситуация, когда при работе над проектом, возникает необходимость отредактировать код, но при этом не потерять работоспособный вариант, который есть на текущий момент. Как правило, создается новая папка, куда копируется продукт, и уже в ней производится работа. Через какое-то время таких папок может стать много, тем более, если над проектом будет работать несколько человек. Такой режим работы вполне оправдает себя, если в итоге получится всего несколько папок-копий проекта. Но, если разработка длится продолжительное время, если есть необходимость частого внесения исправлений, доработок, тестирования новых возможностей, которые в итоге могут не пригодиться, и придется отказаться от таких возможностей, то система контроля версий просто необходима [22].

Прежде всего, она необходима в следующих случаях:

1. Архивация и восстановление. При длительной разработке архивируется какое-то количество состояний проекта. Затем любое из них всегда есть возможность восстановить.
2. Синхронизация работы команды. Очень важно, если разработка ведется не одним человеком, иметь представление о состоянии проекта всем членам команды. При обычном обмене файлами друг с другом велика вероятность, что первый перетрет изменения второго. Сервис контроля версий дает возможность «слить» изменения из разных состояний проекта (его копий) в одно без риска, что чей-то код будет перезаписан чужим.
3. Хранение истории разработки. Часто необходимо знать не только проект в его конечном состоянии, но и то, как он эволюционировал на протяжении всей разработки.

4. Отмена изменений. В том случае, когда правки кода привели к тому, что часть проекта или полностью весь проект перестали работать должным образом, состояние проекта нужно откатывать на рабочую версию. Исправлять все вручную долго и не надежно – можно сделать еще больше ошибок. С помощью системы контроля версий можно быстро и автоматически откатиться на ранние состояния проекта.
5. Альтернативные/экспериментальные реализации. Можно работать над разной реализацией одной части проекта, чтобы проверить, какой из экспериментов окажется лучше. В таких случаях система контроля версий позволит контролировать обе реализации, а после удобно «слить» одну из них в рабочий проект.

Системы контроля версий делятся на две группы: централизованные и распределенные.

Централизованные системы.

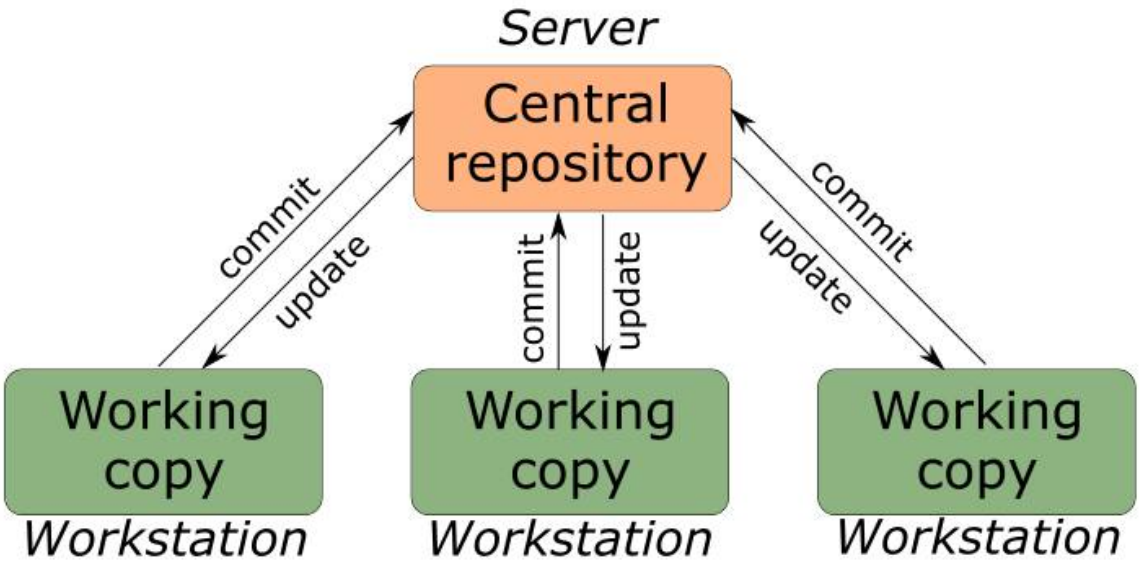


Рис.1 Централизованная система контроля версий

Такие системы представляют собой приложения типа клиент-сервер. Репозиторий проекта хранится в единственном экземпляре на сервере, доступ к которому осуществляется с помощью специального клиентского приложения. Популярным примером такой системы является Subversion (SVN).

К преимуществу централизованной системы можно отнести осведомленность всех разработчиков о том, чем занимается каждый из них и на какой сейчас он стадии разработки.

Серьезным недостатком централизованной системы является единая точка отказа, в связи с использованием централизованного сервера. В случае отказа последнего работа с сервисом контроля версия будет невозможно, как и обмен с другими разработчиками. Если выйдет из строя жесткий диск, на котором хранится центральная база данных, а резервных копий нет, будет потеряна вся история проекта, не считая локальных единичных снимков репозитория.

Распределенные системы.

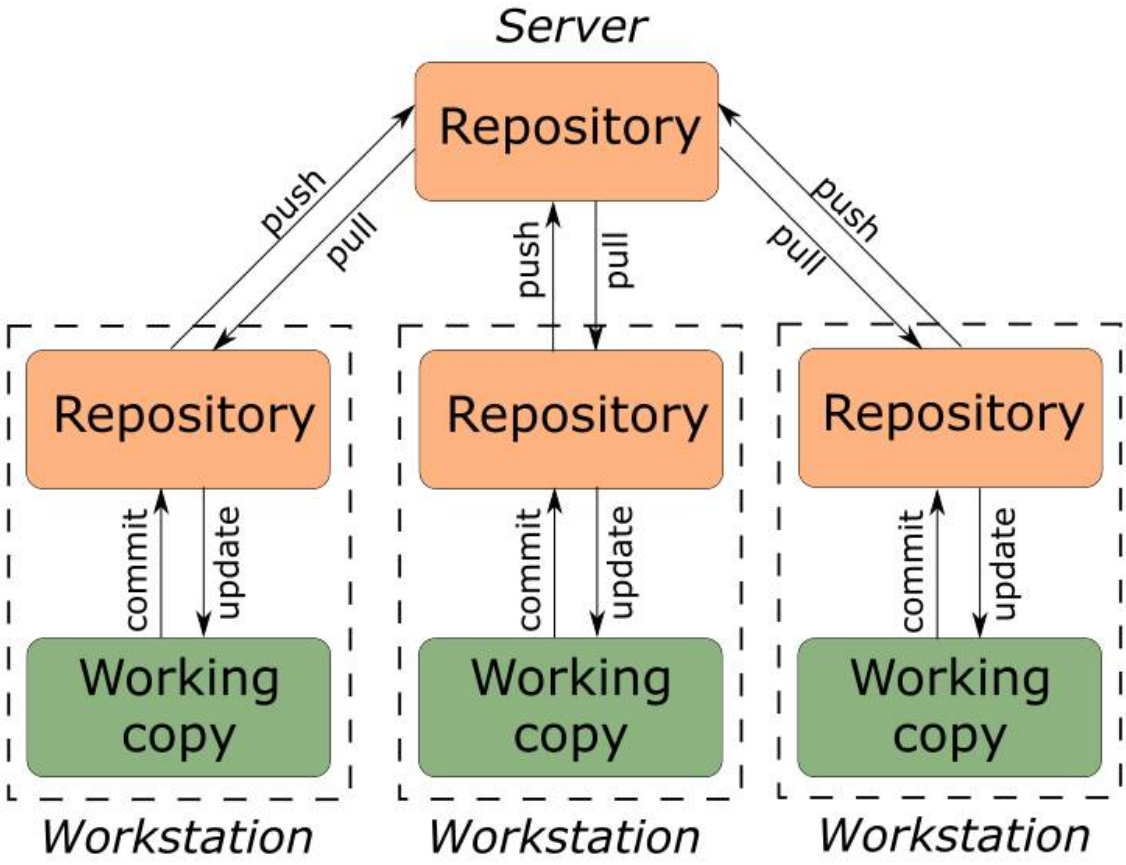


Рис.2 Распределенная система контроля версий

Позволяют хранить копию репозитория у каждого разработчика. Можно выделить центральный репозиторий, где будут собираться изменения из локальных, с ним же локальные репозитории будут синхронизироваться. При работе с распределенной системой, пользователи по мере

необходимости синхронизируются с центральным репозиторием, но работают непосредственно со своей локальной копией.

Большим преимуществом является то, что разработчик является полностью автономным при работе над проектом. Также повышается надежность. Обусловлено это тем, что при потере данных на сервере, можно скопировать клиентский репозиторий на другой сервер и продолжить работу. Ведь каждая копия центрального репозитория представляет собой полную резервную копию всех данных. Распространенной распределенной системой контроля версий можно назвать GIT.

Разработка дизайна. Приложение, которое является не интуитивным и трудным к зрительному восприятию, не способствуем завоеванию большого количества сердец. При разработке дизайна важно учитывать пользовательский опыт. Необходимо построить такую модель, которая будет легка в понимании и освоении всей целевой аудитории, а также не вызывать отторжения и нежелания повторного использования приложения.

Реализация интерфейсов клиент-серверного взаимодействия. Для создания клиентской части веб-приложений обычно используются: HTML, CSS, JavaScript (или VBScript), PHP (или Perl) [10, 14, 17, 18]. Клиентская среда (браузер, например, Google Chrome) является передним краем работы приложения. В этой среде, отображаются страницы приложения. Объекты этой среды, следовательно, обязаны иметь возможность манипулировать страницами. Для этого нужен клиентский язык – JavaScript [15].

Существует огромное количество сред для разработки на Python: Eclipse с плагином PyDev, Eric, PyCharm и другие. Каждая достойна внимания. Конечно, существуют программисты, предпочитающие использовать текстовые редакторы, например: VIM, Gedit, Emacs. Однако, такие решения не подойдут для сложных проектов. Также существует текстовый редактор SublimeText3, имеющий гибкую настройку с помощью огромного количества плагинов, подсветки синтаксиса, темы и цветовые схемы.

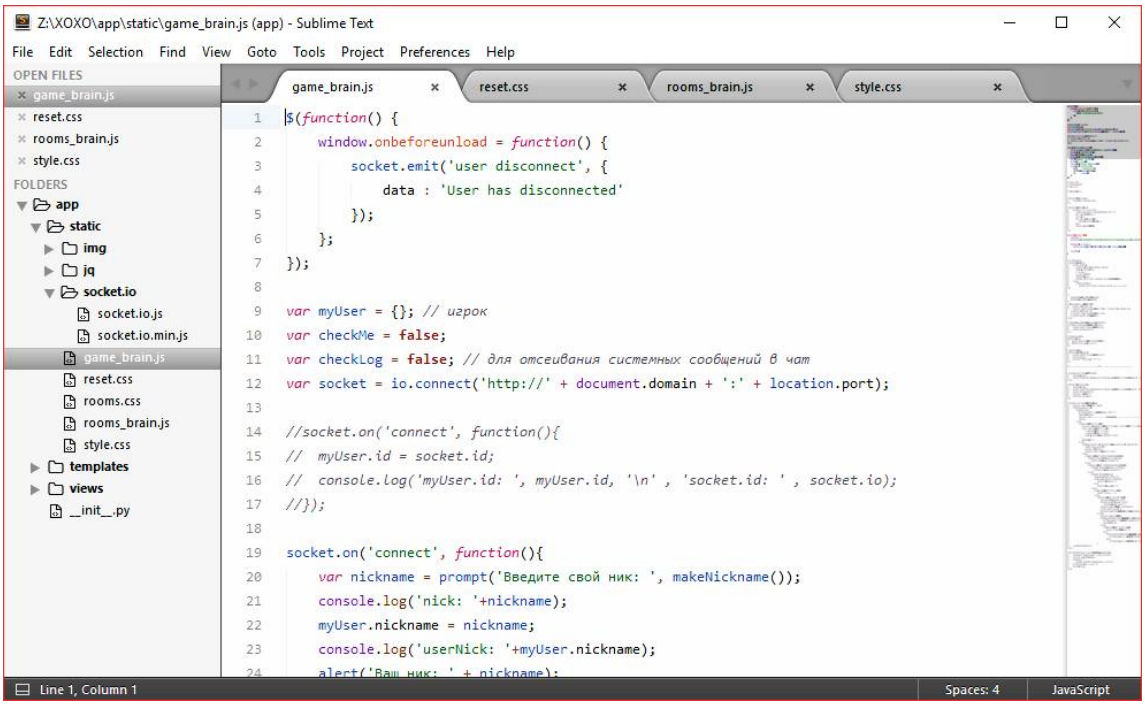


Рис.3 Текстовый редактор Sublime Text

Среда разработки PyCharm имеет такие преимущества, как анализатор кода, режим встроенной отладки (с поддержкой игнорирования библиотечных модулей), инструменты для запуска тестов, поддержка системы управления версиями Git. Стоит отметить наличие Community Edition версии, распространяемой бесплатно, хоть и являющейся коммерческим продуктом.

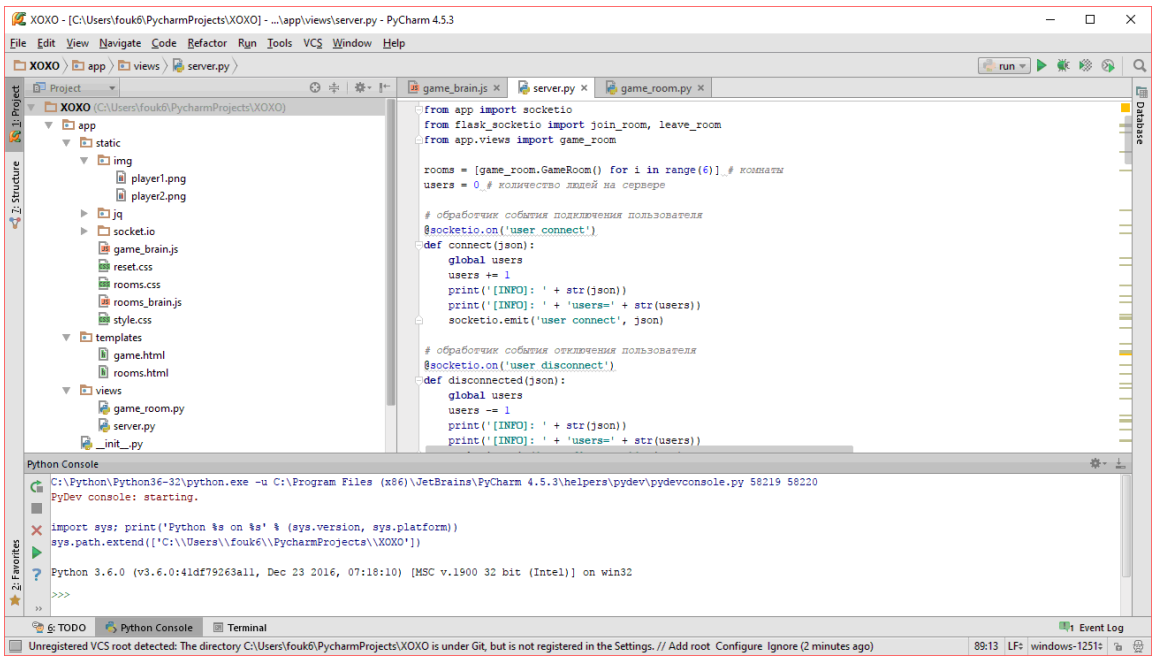


Рис.4 IDE PyCharm

Рассматривая Eclipse, можно отметить, что при выборе этой среды разработки, ее можно будет разумно использовать и в будущих проектах, т.к. Eclipse применяется для написания программ на многих языках программирования. Ну а для результативной работы с Python служит плагин PyDev, добавляя среде достаточно возможностей, далеко не ограничивающихся обычной подсветкой синтаксиса. Данный плагин позволяют пользователю использовать автопродолжение и анализ кода, интерактивную консоль, отладчик и другие функции. Обратной стороной такой гибкости Eclipse можно назвать то, что на него существует бесчисленное количество плагинов, что, несомненно, удобно и позволяет настроить его под практически любую задачу. Но, в то же время, это является поводом и для многочисленной его критики, потому что среда после таких манипуляция и настроек становится «раздута», а ее производительность на посредственных по мощности машинах будет не слишком высока. Как сама IDE, так и плагин распространяются по лицензии Eclipse Public License.

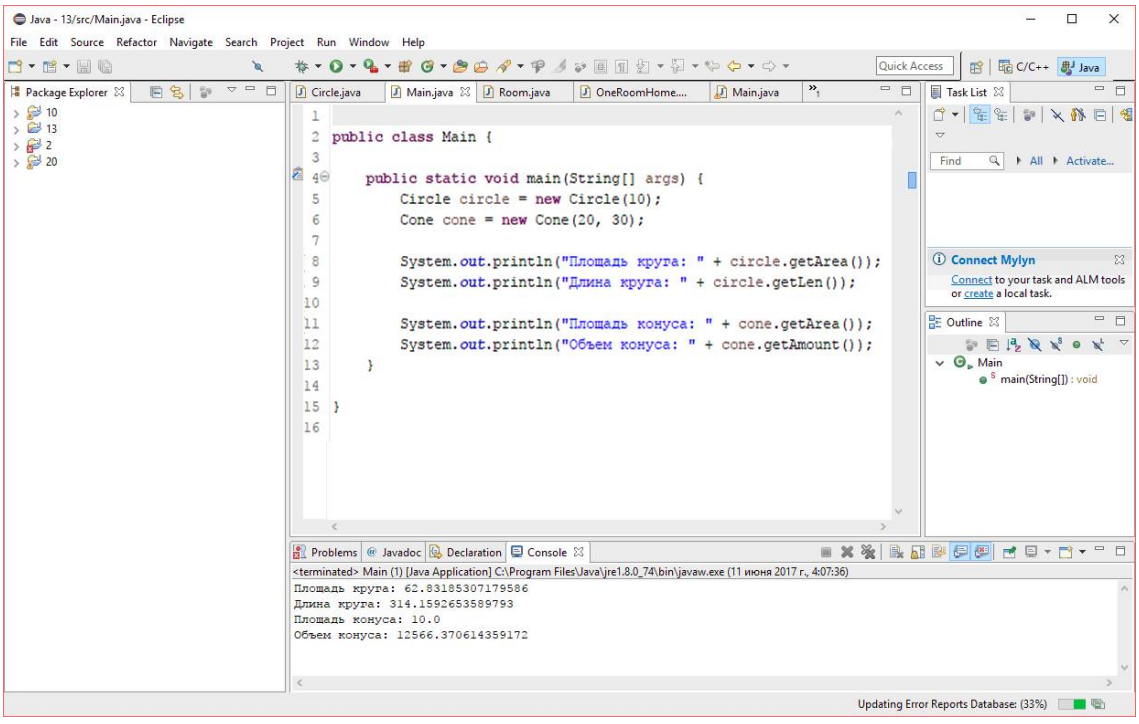


Рис.5 IDE Eclipse

IDE “Eric” включает в себя редактор, интерпретатор языка, графический отладчик и т.п. А модульная строка дает возможностью пользователю не нагружать среду ненужными для его работы компонентами.

Eric имеет поддержку плагинов, которые можно устанавливать непосредственно из рабочей среды. Наличие функции предварительного просмотра формы Qt позволяет эффективно вести разработку Qt-оболочек для приложений. Eric распространяется по лицензии GPL3.

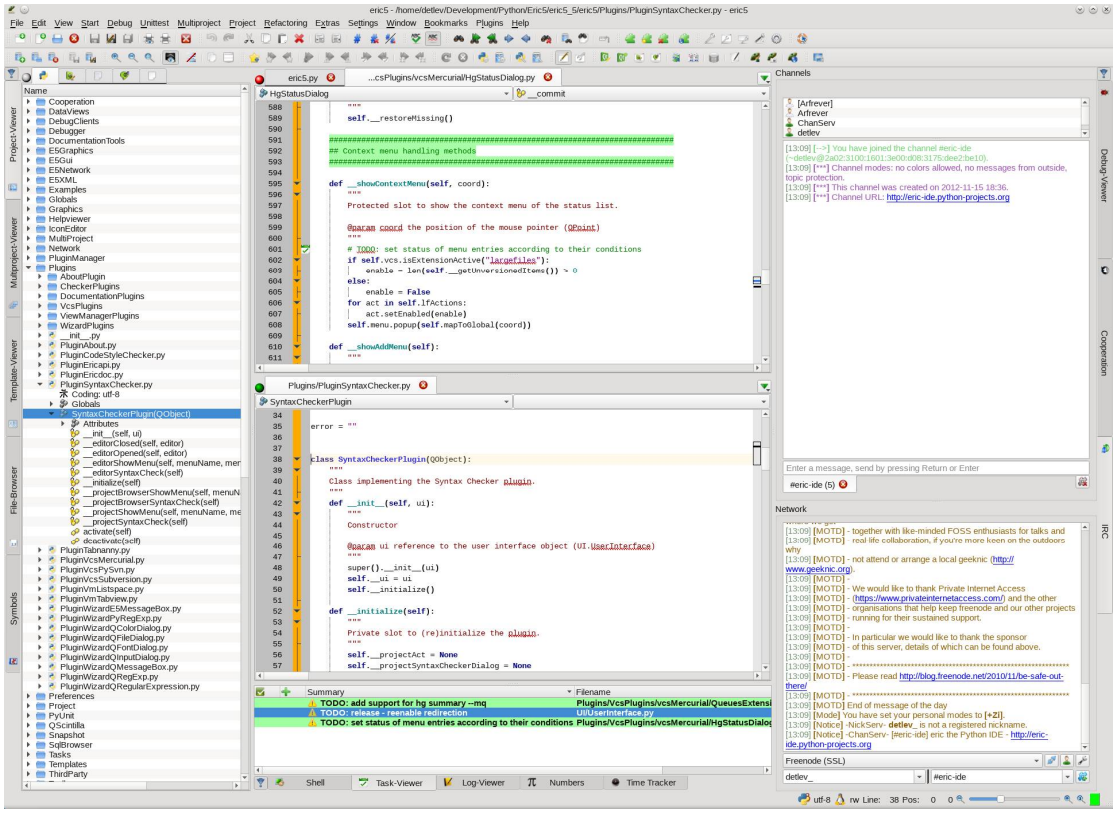


Рис.6 IDE Eric

В большинстве случаев не обойтись без использования сторонних библиотек (сборников программ и объектов, которые используются при разработке программного обеспечения) при разработке продукта. Логичнее (чтобы «не изобретать велосипед») и эффективнее использовать уже готовую библиотеку, протестированную большим количеством ее пользователей.

1.3. Формализованное описание технического задания на разработку игрового клиент-серверного приложения «X O X O»

1. Общие сведения.

1.1. Название: «X O X O».

1.2. Область использования: всенаправленное приложение.

1.3. Данные об авторе: студент группы Б-41, Института математики, информатики и информационных технологий, Рямов И.С.

1.4. Руководитель: ассистент кафедры ИИТиМОИ Алексеевский П.И.

Настоящее техническое задание распространяется на разработку игрового, кроссплатформенного, клиент-серверного приложения, направленного на анализ, обобщение, систематизацию и активное развитие знаний, умений, личностных качеств, а также коммуникацию между пользователями.

2. Основания и назначения разработки.

2.1. Объектом разработки является сетевое компьютерное игровое приложение, использующее кроссплатформенные средства обеспечения клиент-серверного взаимодействия.

2.2. Приложение разрабатывается по личной инициативе автора по согласованию с руководителем выпускной работы, а также в соответствии с учебным планом кафедры.

3. Требования к продукту разработки.

3.1. Аппаратные требования:

3.1.1. Локальный компьютер или смартфон/планшет.

3.1.2. Браузер.

3.1.3. Выход в интернет.

3.2. Указание программного обеспечения, используемого для реализации.

3.2.1. Локальный компьютер с операционной системой Microsoft Windows 10 Pro.

3.2.2. Браузер Google Chrome.

Текстовый редактор Sublime Text 3.

3.2.3. IDE JetBrains PyCharm 4.5.3.

4. Требования к пользовательскому интерфейсу.

4.1. Интуитивно понятный, без необходимости обучения владению приложением.

4.2. Наличие двух страницы для взаимодействия с пользователем.

4.2.1. Страница выбора игровых комнат: название приложения в верхней части страницы, ниже список из шести комнат (2 ряда по 3 комнаты). Под комнатами правила игры.

4.2.2. Страница с игровой комнатой: название приложения в верхней части страницы, в центре – игровое поле, слева от игрового поля – правила игры, справа – игровой чат. В левой верхней части страницы (над правилами игры) кнопки управления «Комнаты» (переход на страницу выбора комнаты) и «Выход» (переход на страницу www.google.com). При входе в игровую комнату пользователю необходимо указать игровое имя. В случае, когда пользователь отменяет это действие, ему присваивается случайно сгенерированное имя из цифр и латинских букв.

5. Состав и содержание работ по созданию приложения.

5.1. Анализ требований к приложению.

5.1.1. Динамическое игровое поле (увеличивается во время игры вертикально вниз на количество строчек, равное отклонению от центральной строчки поля).

5.1.2. Возможность обмена сообщениями между соперниками в игровом чате. Наличие в нем системных сообщений о том, кто зашел в комнату, а кто – вышел.

5.1.3. Наличие интуитивно-понятного интерфейса пользователя.

5.2. Проектирование и разработка приложения.

5.2.1. Разработка технического задания.

5.2.2. Разработка структуры приложения.

5.2.3. Разработка интерфейса.

5.2.4. Реализация приложения.

5.2.5. Написание документации.

5.2.6. Тестирование.

6. Порядок сдачи-приемки продукта.

6.1. Промежуточный контроль – середина марта 2017г., объем – основной функционал, контроль – руководитель.

6.2. Дата отчета руководителю — начало мая 2017г.

Глава 2. СОЗДАНИЕ И ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

2.1. Описание разработки игры

На этом этапе рассмотрим процесс разработки игрового, кроссплатформенного, клиент-серверного приложения «X O X O», служащего для анализа, обобщения, систематизации и активного развития знаний, умений, личностных качеств, а также коммуникации между пользователями.

Разработка приложения началась с анализа и постановки технического задания.

В параграфе 1.2 первой главы были выделены следующие этапы разработки:

1. Цель приложения. Цель разработанного проекта – создание развлекательного приложения с элементами коммуникации. Цель достигается посредством предоставления пользователям функций по взаимодействию друг с другом в игровой форме.
2. Целевая аудитория. Данное приложение создается для людей всех возрастных категорий.
3. Функционал проекта.
   - 3.1. Игровое поле спроектировано как динамический объект. Его размер по вертикали вниз увеличивается при совершении хода игроком в части поля ниже, чем его средняя линия (общее количество строк, разделенное пополам).
   - 3.2. Игровой чат представляет собой обособленный блок коммуникации приложения. Дает информацию пользователям как о сообщениях от оппонента, так и о системных информационных уведомлениях.
   - 3.3. Комнаты для игровых сессий являются списком, который при нажатии на комнату дает понять, есть ли в ней свободное место. Если такого не имеется, оповещает пользователя, что комната полная.

3.4. Наличие встроенного руководства по использованию. Руководство должно включать в себя описание используемой терминологии, правила игры, порядок использования приложения.

Все это позволяет использовать приложение в качестве развлекательного с элементами коммуникации.

4. Выбор языка программирования для веб-сервера. Для проекта был выбран Python, исходя из легкого освоения языка, быстроты написания кода и большого количества необходимых библиотек.
5. Система контроля версий. Для проекта был выбран GIT на основе различий с SVN: изменения в виде метаданных, работа при любых условиях с полноценной локальной копией центрального сервера [19].
6. Разработка дизайна. Приложение имеет простой и понятный для пользователя интерфейс. Цветовая гамма выполнена в мягких контрастных решениях.

Страница выбора игровых комнат совмещает на себе название приложения вверху страницы, упорядоченный список, отображающий названия комнат в центральной части, а также руководство по использованию под списком комнат.

Элементы игровой комнаты сгруппированы следующим образом: вверху – название приложения, в центре – игровое поле, слева – руководство по использованию, справа – игровой чат. Слева сверху размещаются кнопки управления «Комнаты» (переход на страницу выбора комнаты) и «Выход» (переход на пустую страницу браузера www.google.com).

7. Реализация интерфейсов клиент-серверного взаимодействия. В рамках данного проекта были выбраны HTML, CSS и JavaScript. В качестве IDE выбран PyCharm, а текстовым редактором для верстки клиентской части приложения – SublimeText3, основываясь на важных качествах, описанных в параграфе 1.2 первой главы. Из сторонних библиотек выбраны “flask-socketIO” для серверной части. Так как на стороне клиента

может быть использована любая из официальных SocketIO библиотек [12], была выбрана “Socket.IO” для клиентской стороны [20].

2.2. Руководство пользователя по работе с приложением

Рассмотрим **стандартный сценарий пользования приложением**, начиная от входа и заканчивая выходом из него.

Вход в приложение происходит по адресу: http://212.193.77.140:9987/.

После входа на сайт, пользователь имеет возможность выбрать одну из шести комнат, либо ознакомиться с правилами игры.

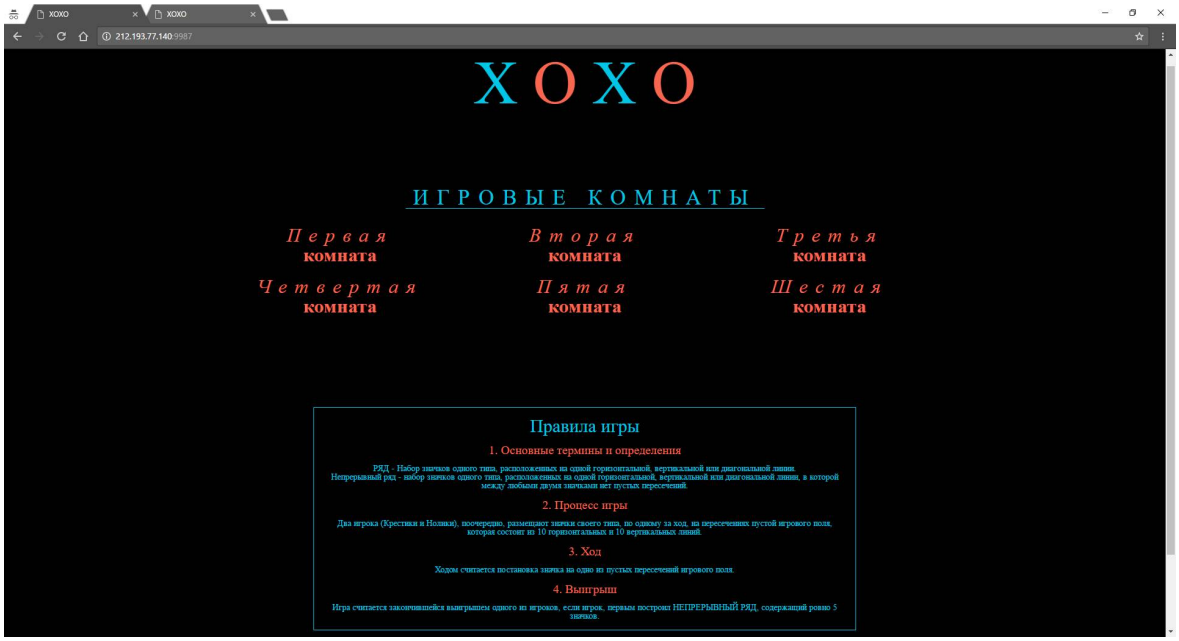


Рис.7 Главное меню приложения

После выбора комнаты у пользователя есть два варианта развития событий:

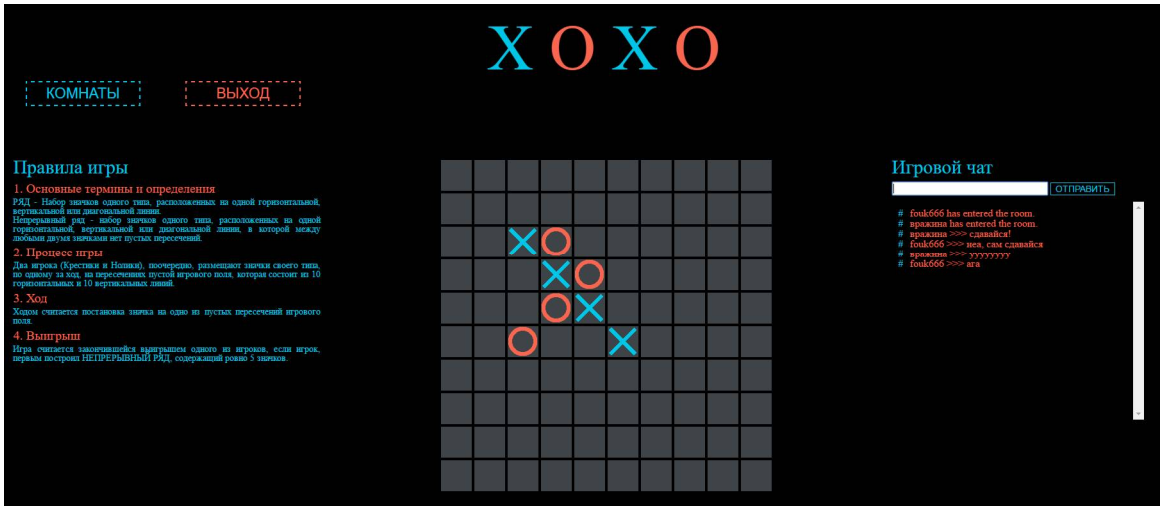


Рис.8 Внутри комнаты

1. В случае, когда в комнате уже находятся 2 игрока, всплывающее информационное сообщение уведомит о том, что комната полная. Пользователь сможет выбрать любую другую комнату или дождаться, пока освободится выбранная изначально.

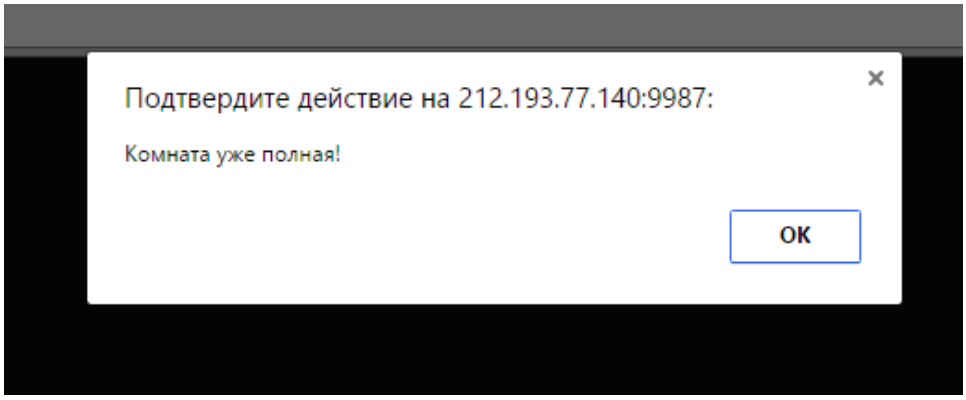


Рис.9 Оповещение о заполненной комнате

2. Если в комнате пусто или в ней находится только 1 игрок, пользователь перейдет на страницу с игровой комнатой.

При входе в комнату необходимо ввести игровое имя. По умолчанию в строке уже будет находиться случайно сгенерированный набор цифр и латинских букв. Если пользователь откажется вводить имя, ему будет присвоено случайно сгенерированное имя.

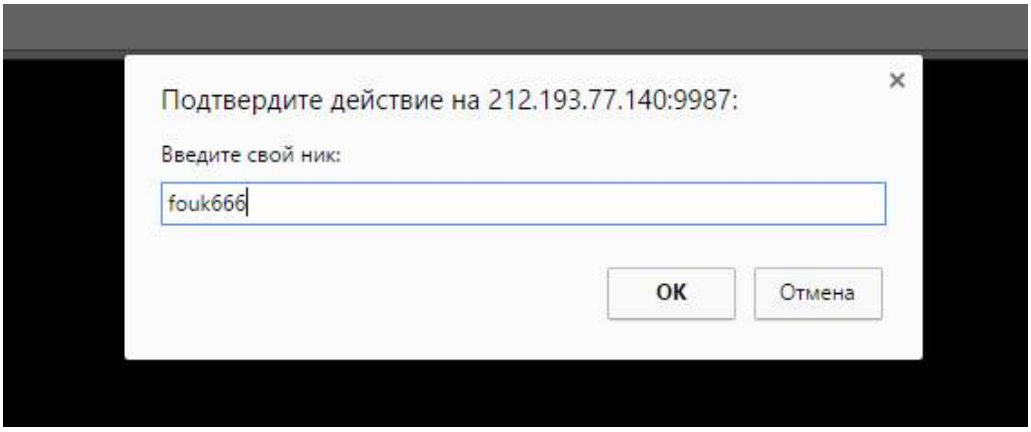


Рис.10 Ввод никнейма при входе в комнату

Вход и выход каждого игрока отмечается в игровом чате комнаты с указанием игрового имени и вида действия.

Рис.11 Логирование событий в чате

Если в комнате находится 1 игрок, совершать ход нельзя, зато можно вспомнить правила игры или написать в чат, но сообщения увидят только те, кто находится в комнате – то есть только один пользователь. Когда присоединится второй игрок, первый сразу поймет об этом, увидев соответствующее оповещение в игровом чате.

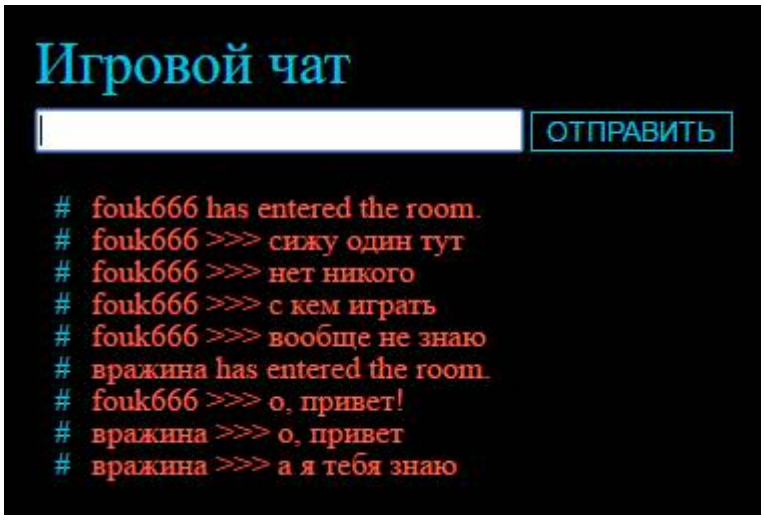


Рис.12 Общение и логирование событий в чате

Начальный ход за тем, кто первый зашел в комнату.

Если во время игры один из игроков по каким-либо причинам не захочет продолжать игру, возможны два варианта развития событий:

1. Выбрать пункт «Комнаты». После этого пользователя перенаправит на страницу выбора комнаты. Второй игрок не останется без внимания – увидит всплывающее информационное сообщение о том, что «кто-то сдался», после также будет перенаправлен на страницу выбора комнаты.

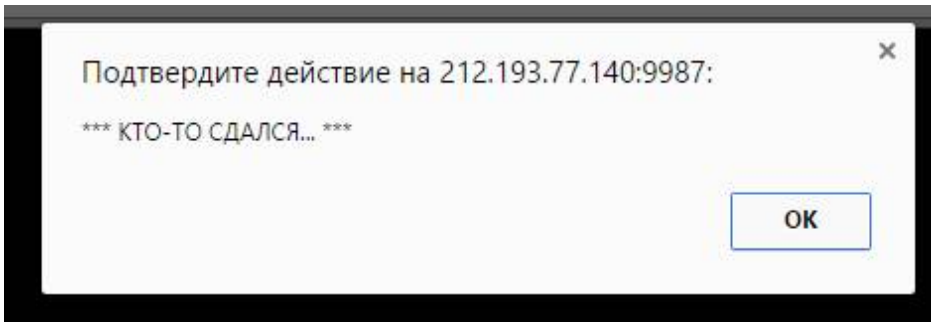


Рис.13 Оповещение при выходе противника из комнаты

2. Выбрать пункт «Выход». После этого пользователя перенаправит на страницу www.google.ru. Второй игрок, как и в случае с выходом в «Комнаты», увидит оповещении о «ком-то сдавшемся» и будет послан на страницу выбора комнаты.

Если игрок совершит ход ниже средней линии игрового поля, последнее увеличится ровно на столько, насколько ниже средней линии будет совершен ход.

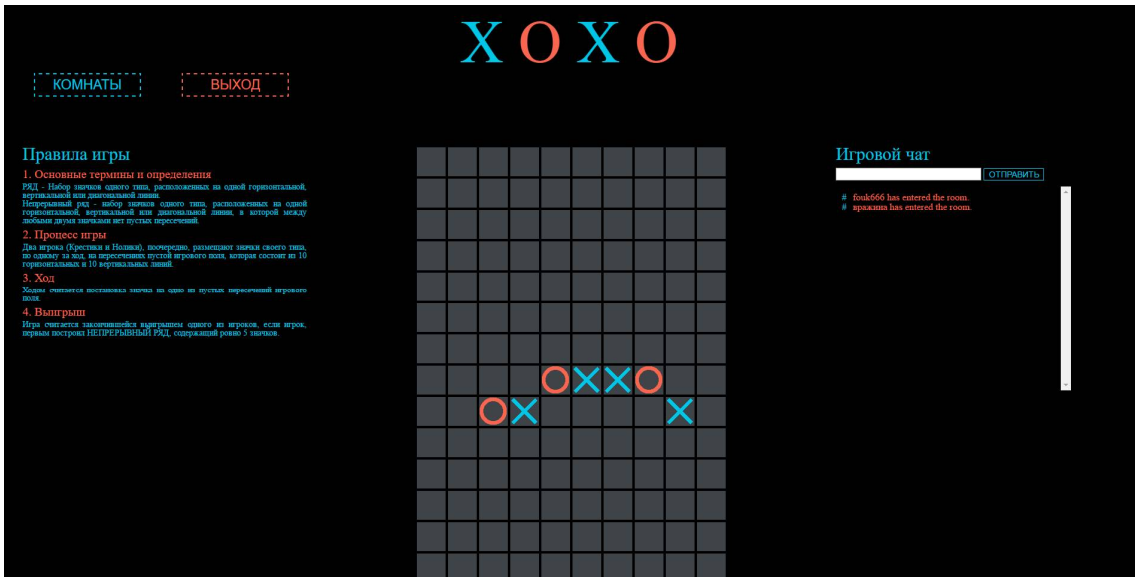


Рис.14 Динамическое игровое поле

При победных условиях, а именно, когда один из игроков составит непрерывный ряд из 5 значков, каждый из пользователей увидит оповещение с соответствующим текстом: победитель – «Ты победил», а проигравший – «Ты проиграл». После игроков перенаправит на страницу выбора комнаты.

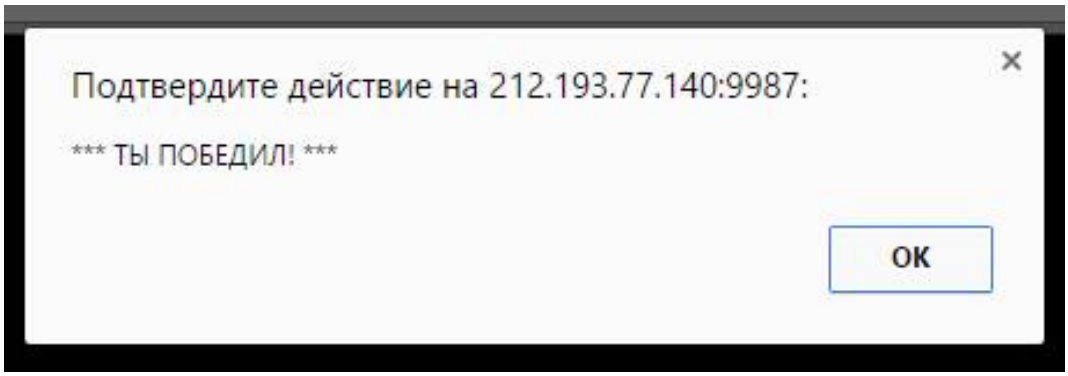


Рис.15 Оповещение о победе

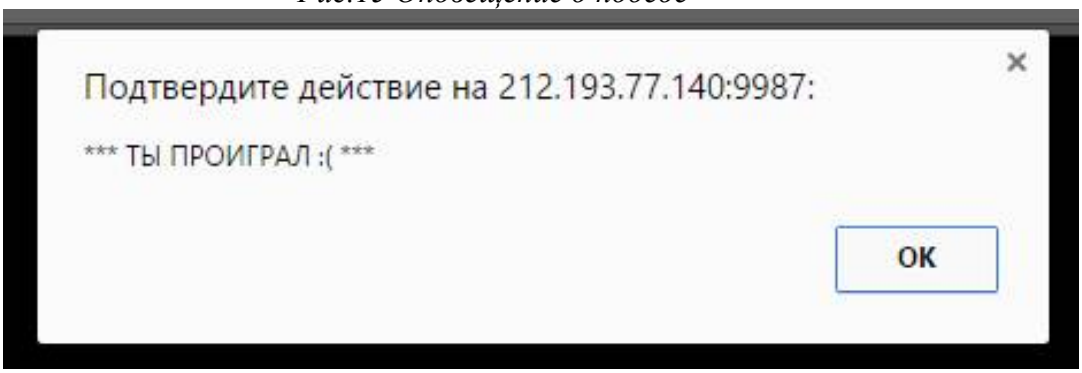


Рис.16 Оповещение о проигрыше

Есть **альтернативный способ запуска приложения** – например, в своей локальной сети развернуть сервер с игрой, когда нет возможности подключиться к удаленному серверу.

Во-первых, для запуска сервера необходим интерпретатор python, который можно скачать по ссылке: python.org/ftp/python/3.6.0/python-3.6.0.exe.

После установки интерпретатора нужно загрузить библиотеки flask и flask-socketio. Чтобы это сделать, надо выполнить в командной строке следующие команды:

1. **pip install flask**
2. **pip install falsk-socketio**

Для запуска локального сервера в командной строке (находясь в корневой директории приложения) выполняем команду:

python routes_local.py

В браузере переходим по адресу http://X.X.X.X:5000, где “X.X.X.X” – ip-адрес машины, на которой запущен локальный сервер.

2.3. Апробация и использование приложения

Была проведена апробация разработанного продукта. Результат апробации помог выявить критические для работы приложения ошибки и недоработки, что позволило исправить их. Также, по итогам апробации были получены результаты анкетирования по продукту.

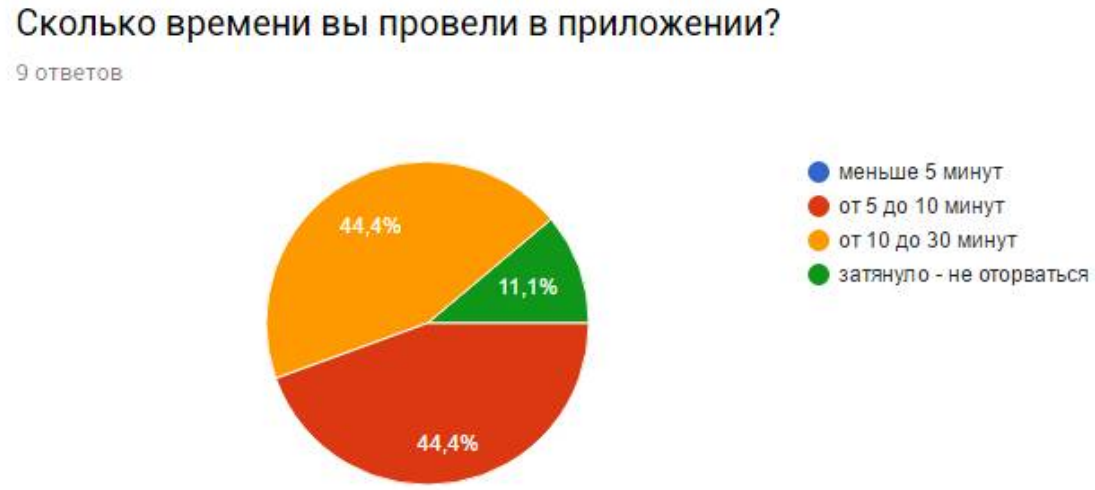


Рис.17 Результат опроса

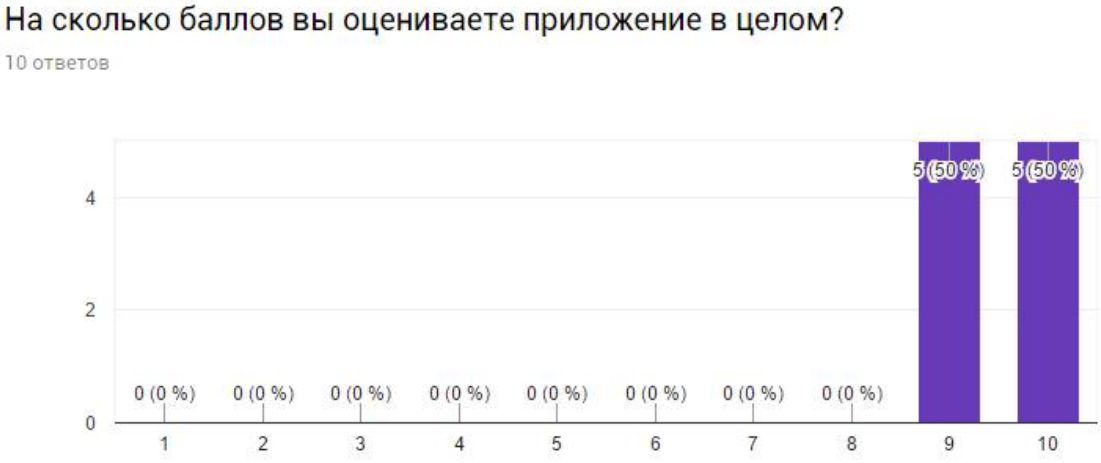


Рис.18 Результат опроса

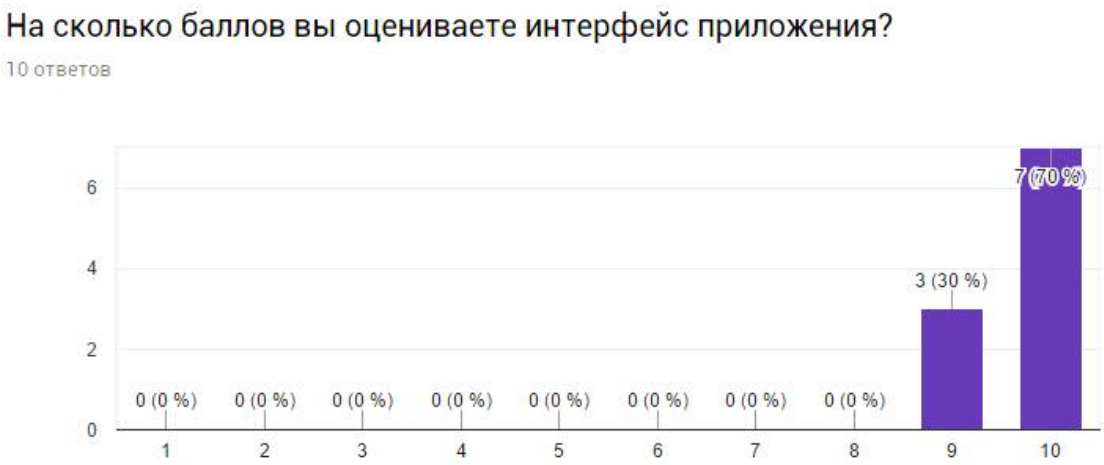


Рис.19 Результат опроса

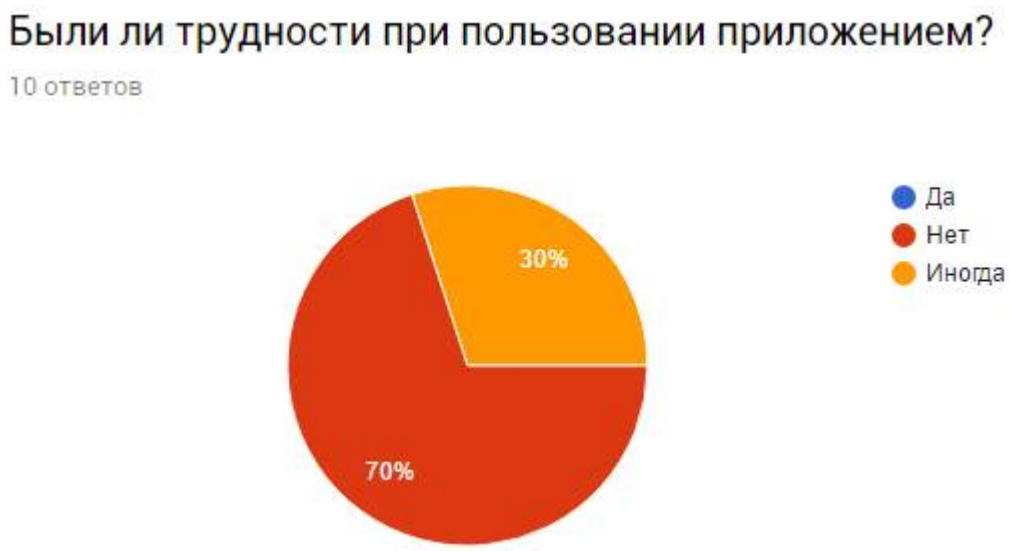


Рис.20 Результат опроса

Рис.21 Результат опроса

Как видно из результатов опроса, приложение было оценено в среднем на 9.5 баллов из 10. В игру было сыграно в среднем от 10 до 30 минут за раз. Дизайн приложения оценен в 9.7 баллов из 10. Общее впечатление от игрового приложения – положительное.

После исправления критических ошибок **приложение осталось в пользовании** небольшой группой постоянных пользователей. Благодаря этому, имеется такая возможность, как постоянное получение обратной связи как о найденных ошибках, так и просто с пожеланиями на будущее, что является большим плюсом для будущей поддержки и развития приложения.

Заключение

В соответствии с целью и задачами, сформулированными в выпускной квалификационной работе, было проделано следующее:

1. Произведен анализ.
2. Выбран подход к разработке.
3. Сформулировано техническое задание.
4. Выбраны инструменты разработки.
5. Разработано и реализовано кроссплатформенное, сетевое, компьютерное, игровое приложение.

Разработанный продукт соответствует всем требованиям технического задания. Произведена апробация и внедрение продукта.

Таким образом, можно утверждать, что цели выпускной квалификационной работа достигнуты, задачи выполнены в полном объеме.

Литература

1. Алексеевский П.И. Обучение программированию студентов на основе методологии унифицированного процесса разработки программного обеспечения // Педагогическое образование в России. 2014. №8. С. 151.
2. Кроссплатформенная и нативная разработка // Студия мобильной разработки Appcraft URL: http://appcraft.pro/portal/cross_platform_vs_native_app/ (дата обращения: 17.02.2017).
3. Основные этапы разработки web-приложений // Education and Science URL: http://www.rusnauka.com/16_ADEN_2011/Informatica/3_85389.doc.htm (дата обращения: 20.03.2017).
4. Сравнение подходов к созданию сайта: проектирование, бриф и agile // Хабрахабр URL: https://habrahabr.ru/post/144442 (дата обращения: 29.03.2017).
5. Стандарты разработки // GeekBrains URL: https://geekbrains.ru/posts/coding_standarts (дата обращения: 23.02.2017).
6. Фреймворк // Словарь электронной коммерции URL: http://emagnat.ru/tag/framework (дата обращения: 18.02.2017).
7. Шапкин С. Компьютерные игры - польза или вред // Психологический журнал, том 20. 1999. №1.
8. Al Sweigart Automate the Boring Stuff with Python: Practical Programming for Total Beginners. - 1st Edition No Starch Press, 2015.
9. Amit Saha Doing Math with Python: Use Programming to Explore Algebra, Statistics, Calculus, and More!. - 1st Edition No Starch Press, 2015.
10. Dive into HTML5 by Mark Pilgrim // Dive Into HTML5 URL: http://diveintohtml5.info/ (дата обращения: 10.03.2017).
11. Eric Matthes Python Crash Course: A Hands-On, Project-Based Introduction to Programming. - 1st edition No Starch Press, 2015.
12. flask-SocketIO // Read the Docs URL: https://flask-socketio.readthedocs.io/en/latest/ (дата обращения: 17.01.2017).
13. Java // Википедия URL: https://ru.wikipedia.org/wiki/Java (дата обращения: 01.04.2017).
14. Jennifer Niederst Robbins Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics. - 4th Edition O'Reilly Media 2012.
15. Jon Duckett JavaScript and JQuery: Interactive Front-End Web Development. - 1st Edition Wiley, 2014.
16. Justin Seitz Black Hat Python: Python Programming for Hackers and Pentesters. - 1st Edition No Starch Press, 2014.
17. Micheal Knapp HTML & CSS: Learn The Fundamentals In 7 days. Independently published 2017.
18. Learn to Code HTML & CSS // Leadership, Product, & Design - Shay Howe URL: http://learn.shayhowe.com/html-css/ (дата обращения: 13.03.2017).

19. Learn Version Control with Git // TOWER URL: https://www.git-tower.com/learn/git/ebook/en/command-line/introduction (дата обращения: 22.03.2017).
20. Socket.IO - Docs // Socket.IO URL: https://socket.io/docs/ (дата обращения: 17.01.2017).
21. Steve Krug Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. - 3rd Edition New Riders, 2014.
22. Система управления версиями // Википедия URL: https://ru.wikipedia.org/wiki/Система_управления_версиями (дата обращения: 15.04.2017).

Приложения

ПРИЛОЖЕНИЕ 1.

Шаблон опроса по приложению

Сколько времени вы провели в приложении? *

- меньше 5 минут
- от 5 до 10 минут
- от 10 до 30 минут
- затянуло - не оторваться

На сколько баллов вы оцениваете приложение в целом? *

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

На сколько баллов вы оцениваете интерфейс приложения? *

1	2	3	4	5	6	7	8	9	10
○	○	○	○	○	○	○	○	○	○

Были ли трудности при пользовании приложением? *

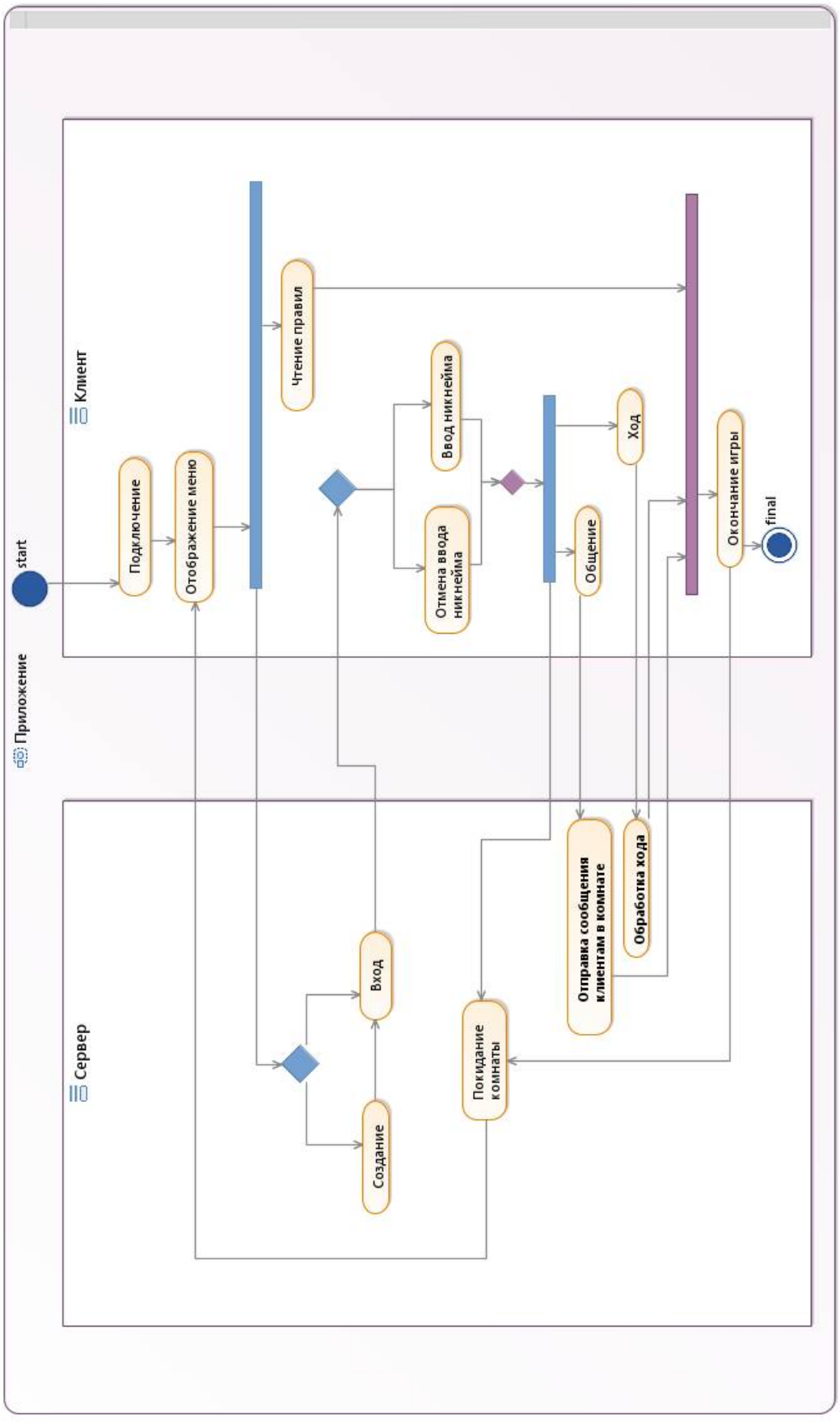
- Да
- Нет
- Иногда

Что в будущем вы хотели бы видеть в приложении? *

Развернутый ответ

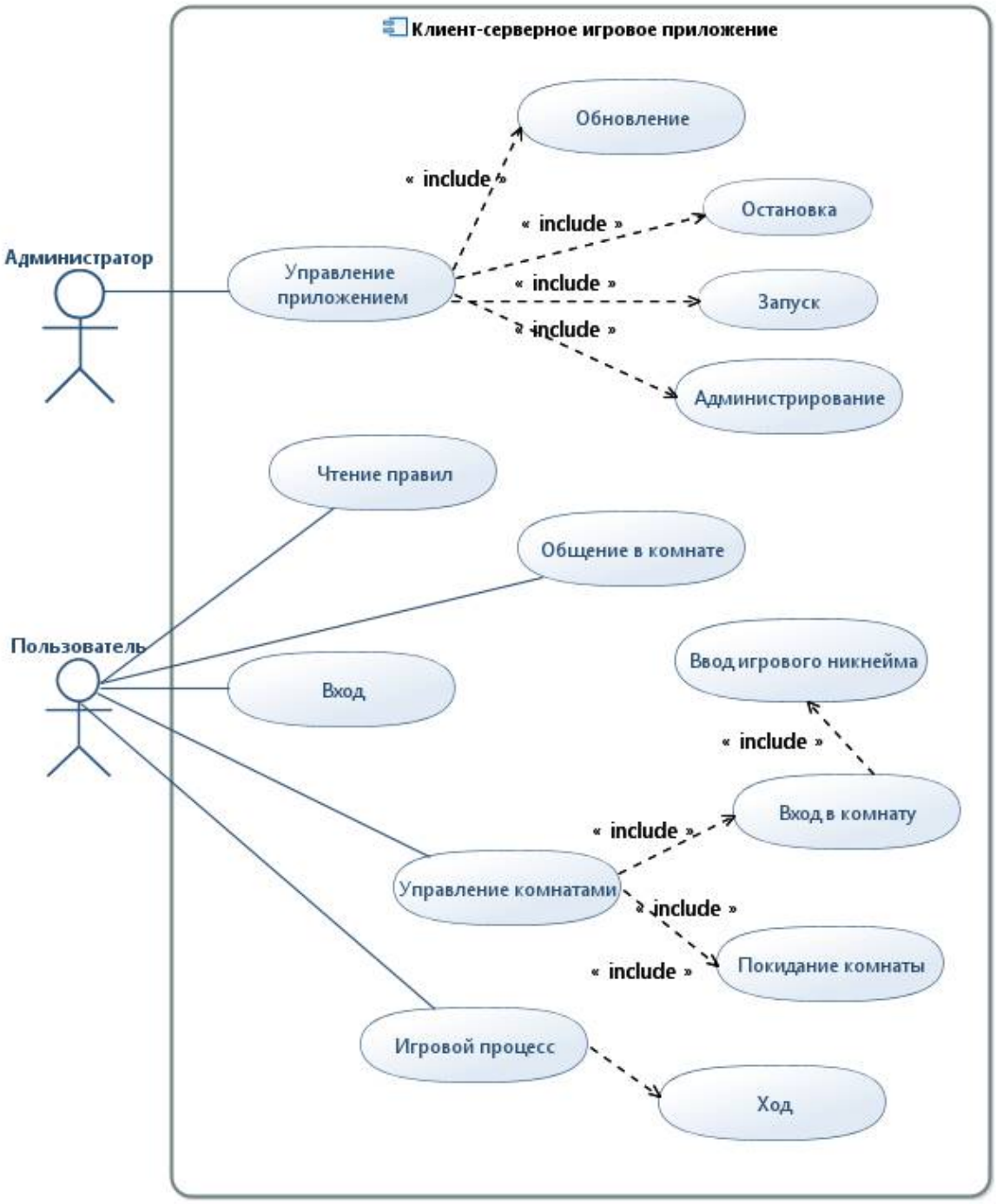
ПРИЛОЖЕНИЕ 2.

Диаграмма деятельности



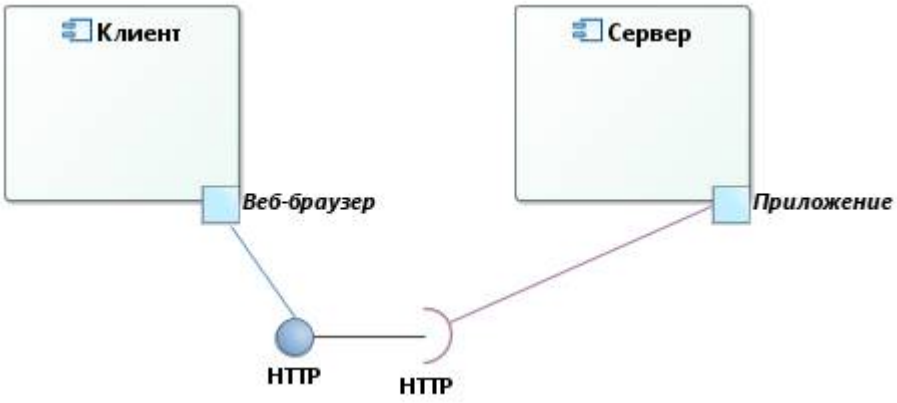
ПРИЛОЖЕНИЕ 3.

Диаграмма прецедентов



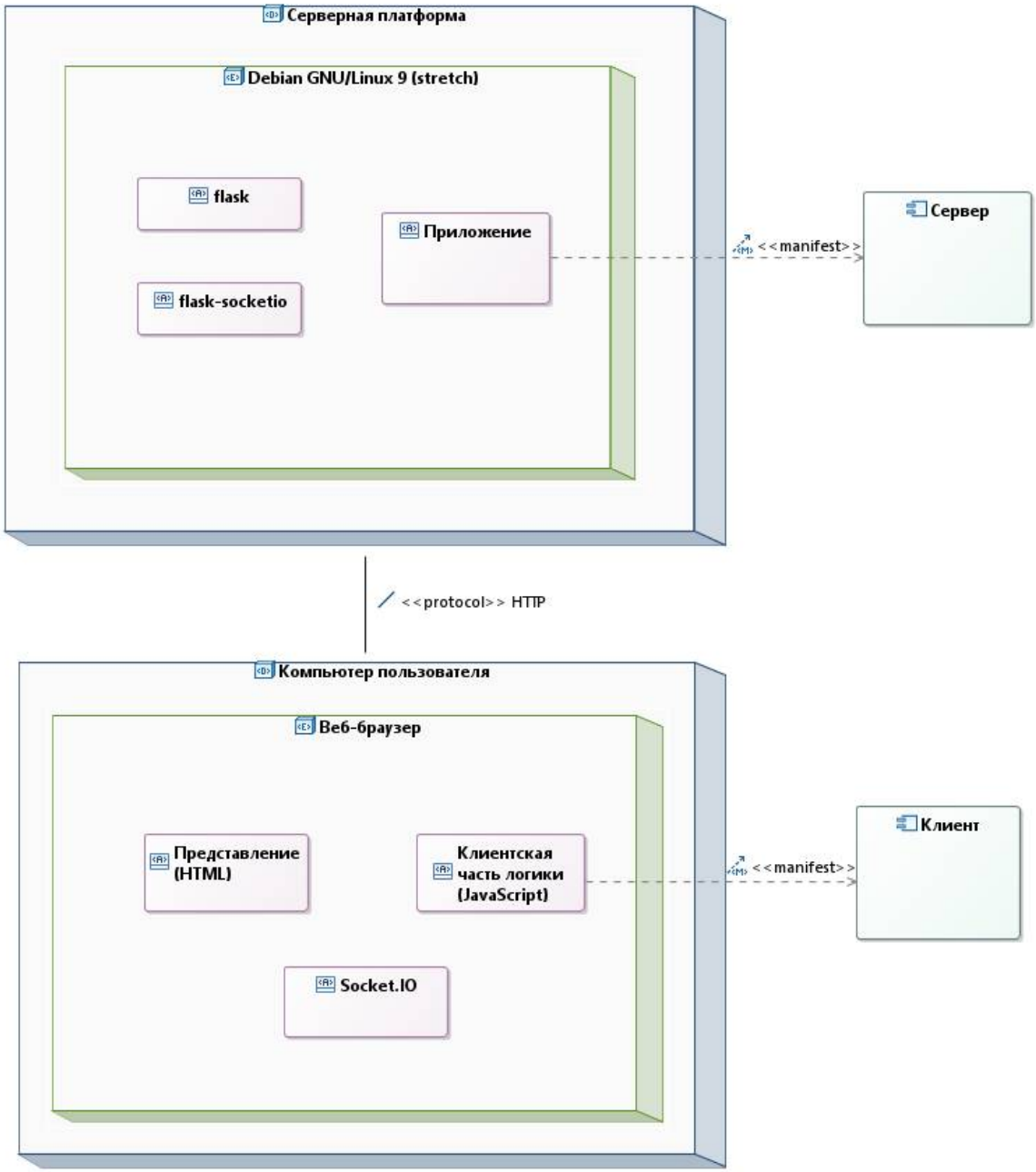
ПРИЛОЖЕНИЕ 4.

Диаграмма компонентов



ПРИЛОЖЕНИЕ 5.

Диаграмма развертывания



ПРИЛОЖЕНИЕ 6.

Диаграмма классов